西南科技大学

# 网络攻击与防御
# 实验报告

实验题目：ICMP 重定向攻击

学生姓名：

学生学号：

## 一、 实验作业题目

1. ICMP 重定向攻击

2. ICMP 中间人攻击

## 二、 实验思路

容器启动后，查看容器信息

```
[03/13/25]seed@VM:~/.../Labsetup$ dockps
d63abef2e78c  host-192.168.60.6
5c59e3b7cbcb  attacker-10.9.0.105
7d57fda367c4  malicious-router-10.9.0.111
6a9fc34cee6b  victim-10.9.0.5
edfb13bc1989  router
4b0b20cf68e8  host-192.168.60.5
```

### ICMP 重定向攻击

确保受害机开启接受 IPv4 的重定向功能

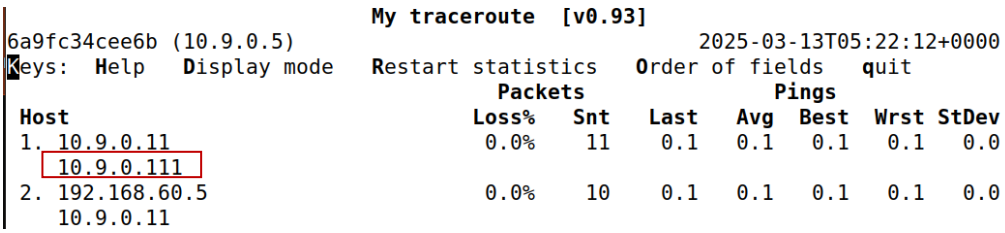
在受害者容器上运行 ip route，可以看到目的地址在 192.168.0.0/24 的数据包，将通过 10.9.0.11 这个网关发出

```
root@6a9fc34cee6b:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
```

执行 mtr -n 192.168.60.5 跟踪数据包到 192.168.60.5 所经过的路由节点，可以看到经过了 10.9.0.11 网关，验证了上述路由。

```
                             My traceroute  [v0.93]
6a9fc34cee6b (10.9.0.5)                               2025-03-13T05:19:00+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                 Packets                Pings
 Host                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                   0.0%     5    0.1   0.1   0.1   0.3   0.1
 2. 192.168.60.5                0.0%     5    0.1   0.1   0.1   0.1   0.0
```

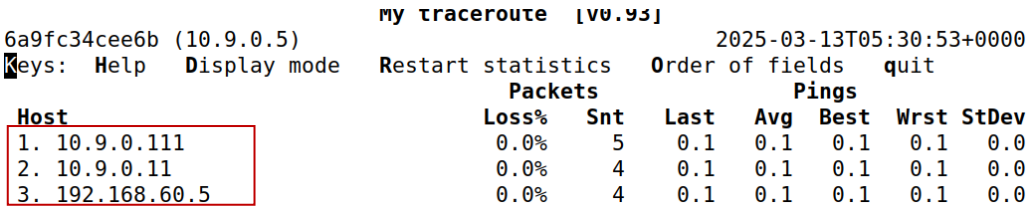
在 attack 中运行脚本 icmp_direct.py，受害者的终端发生变化




观察代码可以知道构造并发送了一个ICMP 重定向消息数据包，告知受害机下次发往 10.9.0.11 的数据包应通过 10.9.0.111。

由于受害机开启了接受 IPv4 的重定向功能，所以接受了这个重定向信息，从而改变了数据包的路由路径。

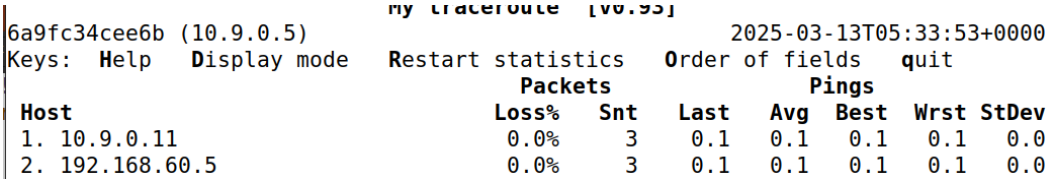
完成重定向攻击后再次使用 mtr 查看




可以看到发送的 ICMP 包先去了 10.9.0.111 再去了 10.9.0.11 最后才到达 192.168.60.5

清理缓存内容后重定向攻击失效，说明重定向攻击依赖于路由缓存来维持其效果。

```
root@6a9fc34cee6b:/# ip route flush cache
root@6a9fc34cee6b:/# ip route show cache
```




防御方法：

关闭 IPv4 的重定向功能，net.ipv4.conf.all.accept_redirects=0

定期清理路由缓存

只接受来自可信网关或特定网络区域的重定向消息，拒绝来自其

他未知或不可信源的重定向请求。

## ICMP 中间人攻击

关闭路由器的 IP 转发功能后，建立受害机和 host（192.168.60.5）间的 TCP 连接，然后运行 mitm_sample.py 脚本。

在受害机中输入 seedlabs 后，host 会变成 AAAAAAA

```
root@6a9fc34cee6b:/# nc 192.168.60.5 9090
111
seedlabs
```

```
[03/13/25]seed@VM:~/.../Labsetup$ docksh 4
root@4b0b20cf68e8:/# nc -l -p 9090
111
AAAAAAAA
```

说明攻击成功，实现了对通信内容的篡改

```
.
Sent 1 packets.
*** b'AAAAAAAA\n', length: 9
.
Sent 1 packets.
*** b'AAAAAAAA\n', length: 9
.
Sent 1 packets.
*** b'AAAAAAAA\n', length: 9
.
Sent 1 packets.
*** b'AAAAAAAA\n', length: 9
.
Sent 1 packets.
*** b'AAAAAAAA\n', length: 9
```

在运行脚本处发现有大量的 ICMP 被篡改，因为这个代码捕获了所有的 TCP 数据包，包括程序自己生成的数据包。

更改过滤器，将过滤条件改为“tcp and inbound”，使得它不会捕获自己的数据包

```
f = 'tcp and inbound'
pkt = sniff(iface='eth0', filter=f, prn=spoof pkt)
```

```
^Croot@7d57fda367c4:/volumes# ./mitm_sample.py
LAUNCHING MITM ATTACK.........
*** b'seedlabs\n', length: 9
.
Sent 1 packets.
```

防御方法：

配置防火墙，只允许来自可信源的数据包通过，限制对特定端口和服务的访问。

在建立 TCP 连接时，对连接的合法性进行验证，检查源 IP 地址、目的 IP 地址、端口号等信息是否符合预期

启用 IPsec 加密，IPsec 可以对 IP 数据包进行加密和认证，确保数据包在传输过程中的完整性和保密性。

在应用程序中对用户输入进行验证和过滤，防止恶意输入被篡改。

## 三、 实验分析及总结（写自己在实践过程中遇到的**问题及解决方法**；本次实验**体会、收获**）

问题： 更改过滤器，如何使得它不会捕获自己的数据包

解决方法：一开始排除源 IP 或目的 IP 为本地主机 IP 的数据包，但并不起作用。通过将过滤条件改为 “tcp and inbound”，可以只捕获进入网络接口的 TCP 数据包，避免捕获自己生成的数据包。

体会及收获：

在本次 ICMP 攻击实验中，我收获颇丰。通过 ICMP 重定向攻击与中间人攻击实验，深入理解了网络攻击原理与数据包的处理机制。在实践过程中，遇到过滤器设置难题，尝试排除本地 IP 地址失败，最终通过“tcp and inbound”成功解决。

这不仅提升了我解决技术难题的能力，还让我认识到网络安全的复杂性与重要性。明白了看似简单的网络配置，一旦被恶意利用，就可能带来严重后果。